Web Application Security

Implementing the Superstition in JDeveloper

Peter Koletzke
Technical Director & Principal Instructor

Duncan Mills
Java Evangelist & Consulting Product Manager

Believe It or Not

> Security is mostly a superstition.
> It does not exist in nature,
> nor do the children of men
> as a whole experience it.
> Avoiding danger is no safer
> in the long run than outright exposure.
> Life is either a daring adventure
> or nothing.
>
> —*Helen Keller (1880-1968)*

Survey

- Jobs
  - Developer?
  - DBA?
  - Sys admin, others?
- Web Application Work
  - J2EE?
  - .NET?
  - PHP, ColdFusion, others?
- Tools
  - JDeveloper
  - Eclipse
  - Others

Agenda

- Why security?
- OC4J security
- Set up the user repository
- Set up web descriptor security
- Set up View layer security

Slides and white paper with sample code will be available on the Quovera and NoCOUG websites

Application Areas of Exposure

- Unapproved users can run the application
- Approved users can access data they should not access
  - Access through View or Model code
- You cannot track who accesses the data
  - Approved or not
- Users bend normal query functions to gain unauthorized access
  - SQL injection

Security Objectives

- Ultimate security may just be superstition, however, data must be protected
- Why is exposure greater in web apps?
  - More accessible to any WWW hacker than an internal app
  - Given time and CPU power, a motivated hacker can break any security scheme
- Main objective with any security system:
  - Make breaking in as difficult as possible
- Assume file system of app server is secure
  - Reading configuration files with user identity and application security should be really difficult
  - Operating system and network has other security needs and features

Two Primary Operations

- Authentication
  - Validate that the user is who she/he claims to be
    - Normally done with passwords
    - With extra equipment, could be something else
      - Retinal scan, thumbprint, DNA (?)
- Authorization
  - Allow authenticated user access to specific resources
  - Usually done with security roles
    - Like database roles
    - Application components (pages, functions) and data are made available to named roles
    - Users are enrolled in roles
      - User has access to whatever the role is granted

Agenda

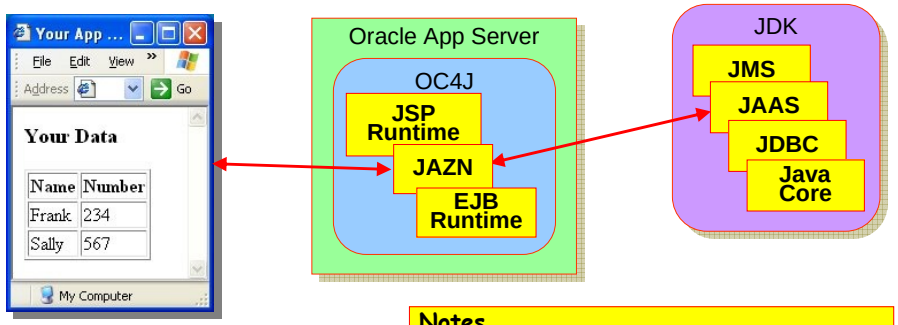
- Why security?
- OC4J security
- Set up the user repository
- Set up web descriptor security
- Set up View layer security

How to Implement the Superstition

- Use recognized, prebuilt, proven, supported security technologies
- Java Authentication and Authorization Services (JAAS)
  - Java API library in the J2SE Development Kit (JDK or J2SDK))
- One solution: JAZN
  - Available in Oracle App Server Containers for J2EE (OC4J)
    - Oracle Application Server's J2EE runtime
  - Java authorization and authentication
  - An API to JAAS
    - Meta-API?
  - You configure your application to use JAZN

Summarizing That

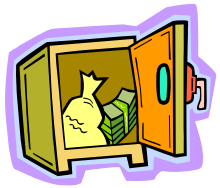
- OC4J in Oracle App Server contains JAZN that calls JAAS in the JDK



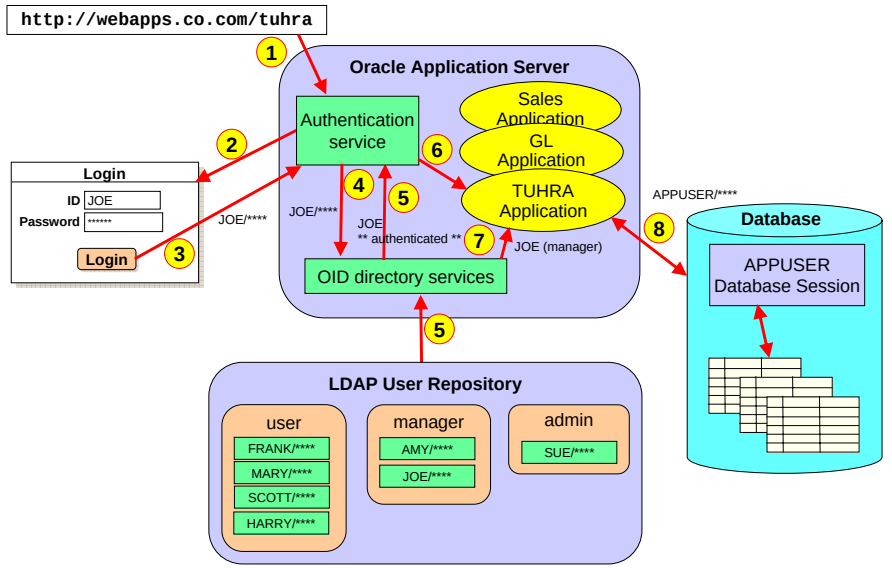
Notes
- This is only one method for security.
- This is not to scale.

The User Repository

- The storehouse of user and role information
  - A.k.a., *credentials store* or *identity store*
- JAZN can tap two types of user repositories
  - XML
    - Extensible Markup Language
    - Properties file containing user and role definitions
    - With 10.1.3 OC4J, can set up lightweight SSO
  - LDAP
    - Lightweight Directory Access Protocol
    - A communications protocol
    - Oracle Internet Directory (OID)
      - Used for Single Sign-On (SSO)
    - OID can read other LDAP providers
      - E.g., Microsoft Active Directory

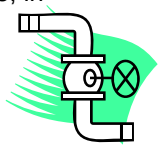


Application Security Flow



Application Security Flow

1. User sends HTTP request including a context root indicating a particular application.
2. The authentication service determines the method (XML or LDAP) and presents a login page.
3. The user enters an ID and password and submits the login page.
4. The authentication service requests OID to verify the user and password.
5. OID verifies the password in from the LDAP source and indicates pass or fail to the authentication service.
6. The authentication service accesses the application and places the user name into the HTTP session state.
7. The application can request the username or group (role, in this example, “manager”) to which the user belongs
8. The application connects to the database using the application database user account (APPUSER) written into a configuration file.



Agenda

- Why security?
- OC4J security
- Set up the user repository
- Set up web descriptor security
- Set up View layer security

Application Security Tasks

Administrator

- ✓ Select a security system
  - JAZN here
- Set up user repository roles and users
- Enroll users in roles in the user repository
- Switch user repositories
  - Before production

Developer

- Set up logical application roles (used in application)
- Configure login method for the application
- Protect pages based on roles
- Protect items based on roles
- Display the logged-in user
- Secure Model attributes

JDeveloper Support

- Define these files using JDeveloper’s XML property editors
  - **`<appname>-jazn-data.xml`**
  - **`<appname>-oc4j-app-data.xml`**
  - **`web.xml`**
  - These files configure the Embedded OC4J Server in JDeveloper
- “<appname>” is the application workspace name in JDeveloper
  - Transfer these settings to the “system” level files in the 10.1.3 server
    - **`system-jazn-data.xml`**
    - **`system-oc4j-app-data.xml`**

Set Up Roles and User Accounts

- For XML provider in **<appname>-jazn-data.xml**
- Define within a realm (namespace within the XML file)
  - By default jazn.com

Role

```
<role>
  <name>admin</name>
</role>
```

User

```
<users>
  <user>
    <name>SKING</name>
    <credentials>{903}1JHgZuUDp..
    </credentials>
  </user>
</users>
```

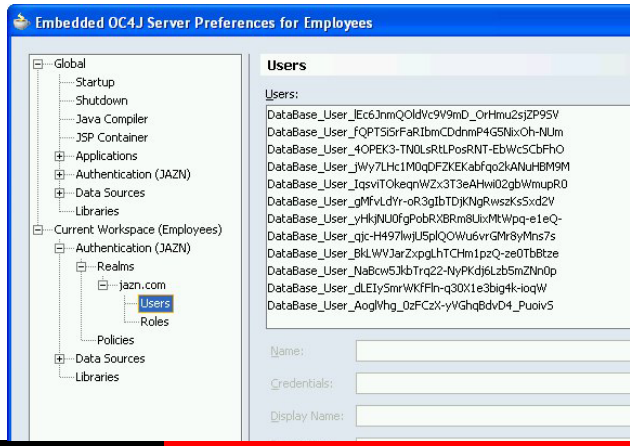
password obfuscation

Users in Role

```
<role>
  <name>admin</name>
  <members>
    <member>
      <type>user</type>
      <name>SKING</name>
    </member>
    <member>
      <type>user</type>
      <name>AHUNOLD</name>
    </member>
  </members>
</role>
```

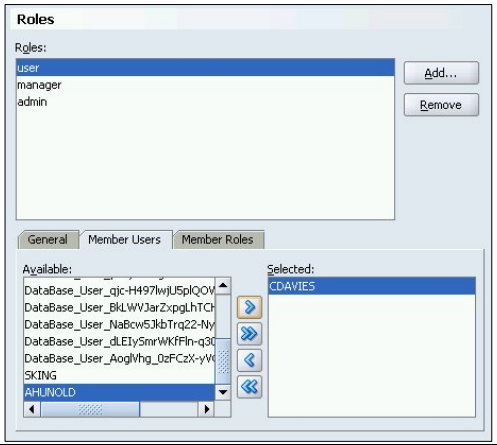

Users and Roles in JDeveloper

- **Tools | Embedded OC4J Preferences** after selecting the application
  - Current Workspace\Authentication\ realms\jazn.com
- Users node
  - Click Add
  - Define name and password
  - Password is obfuscated
- Roles
  - Click Add
  - Enter name, description



Enroll Users in Roles

- Members Users tab on Roles page
  - Shuttle users to Selected area.



Demo

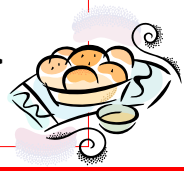
Agenda

- Why security?
- OC4J security
- Set up the user repository
- Set up web descriptor security
- Set up View layer security

Set Up Logical Application Roles

- In **web.xml** (web application deployment descriptor)
- Standard J2EE XML file – standard contents
- Abstracts the roles required by the application from the user repository roles

```
<security-role>
  <description>Administrative users</description>
  <role-name>admin</role-name>
</security-role>
<security-role>
  <description>Management users</description>
  <role-name>manager</role-name>
</security-role>
```



Logical Application Roles

- On web.xml node in ViewController\Web Content\WEB-INF, select **Properties**
  - Web Application Deployment Descriptor dialog
  - On Security Roles page, click Add

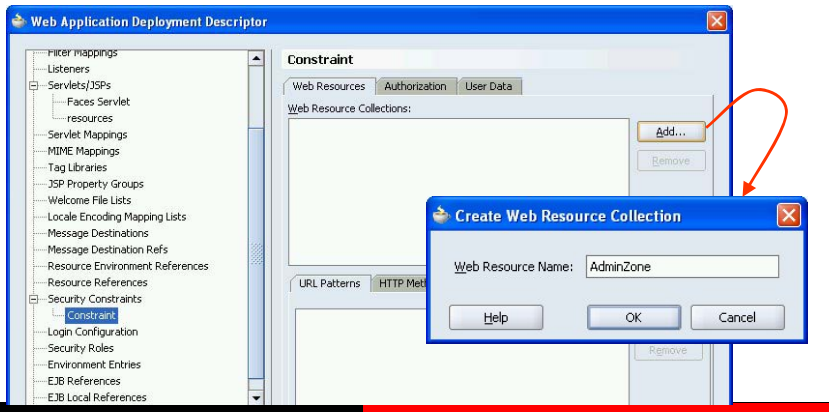
Define Security Constraints

- Used to map logical roles to URL patterns
- Restricts access to a set of files based on role
- URL pattern represents a directory and file names

```
<security-constraint>
  <web-resource-collection>
    <web-resource-name>UserZone</web-resource-name>
    <url-pattern>faces/pages/*</url-pattern>
  </web-resource-collection>
  <auth-constraint>
    <role-name>user</role-name>
    <role-name>admin</role-name>
    <role-name>manager</role-name>
  </auth-constraint>
</security-constraint>
```

Security Constraints

- On Security Constraints node (web.xml), click New
  - A Constraint child node will appear
- Click Add and name the constraint
  - Order matters - start with most restrictive



Define the Constraint

- Select Web Resource Collection (AdminZone)
  - On Authorization tab, select the roles
  - These roles will be constrained to the URL patterns you define next
- On Web Resources tab, select collection
  - Click Add and Enter path and file names (or "*" for all)
- Repeat creation of constraint for all other URL patterns needed
  - E.g., UserZone constraint for "faces/pages/*" URL pattern
- Set *Redirect* on nav. case to "true"
  - That way, the browser will request the page using the URL pattern

Constraint Gotcha

- Top level directories defined as URL patterns may override lower level directories
  - E.g., URL patterns for "/faces/*" and "/faces/admin/*"
    - User role assigned "/faces/*"
    - Admin role assigned "/faces/admin/*"
  - Admin pattern is more restrictive and defined first in constraints
  - "/faces/*" pattern may allow users to access admin pages
- Solution: define a public directory so pattern for user role is "/faces/public/*"

Define Application Login

- Set login method
  - Basic or form-based authentication
  - Set in **`web.xml`**

Basic

```
<login-config>
  <auth-method>BASIC</auth-method>
</login-config>
```

Form-based

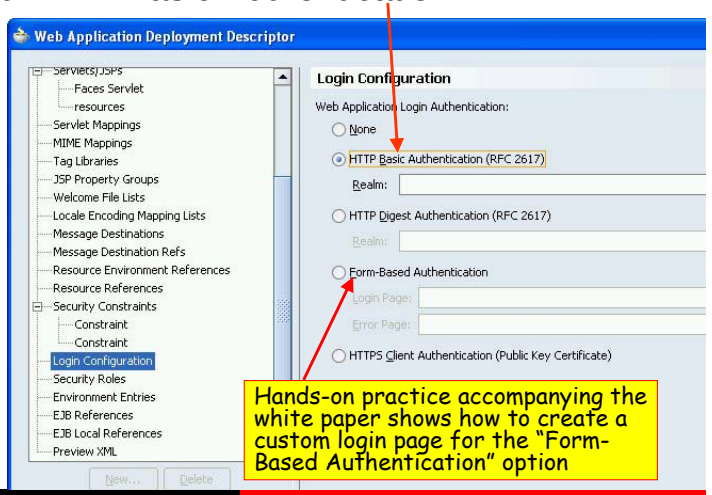
```
<login-config>
  <auth-method>FORM</auth-method>
  <form-login-config>
    <form-login-page>security/login.jsp</form-login-page>
    <form-error-page>security/login.jsp</form-error-page>
  </form-login-config>
</login-config>
```

Specify a login and error page.

Demo

Define Login Method

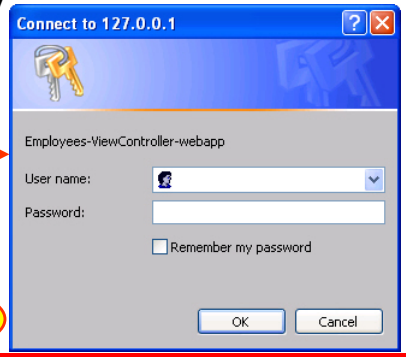
- Login Configuration page (web.xml)
  - Select HTTP Basic Authentication



Hands-on practice accompanying the white paper shows how to create a custom login page for the "Form-Based Authentication" option

Testing Basic Authentication

- Reminder:
  - admin can access faces/pages/admin/*
  - user and admin can access faces/pages/*
- Define pages for admin and user
  - One page in each directory
- Test each page
- Basic authentication dialog will appear when you run the page
- Test password protection



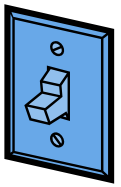
Demo

Switching User Repositories

- XML user repository is handy for development
  - Stored in **<appname>-jazn-data.xml** in the application root directory – edit it manually
  - Can manage this locally for application development
- LDAP is used for enterprise production systems
- Switch it in **<appname>-oc4j-app-data.xml**

From:
```
<jazn provider="XML"
      location="jazn-data.xml"
      default-realm="jazn.com"/>
```

To:
```
<jazn provider="LDAP"
      location=
      "ldap://ldap.tuhra.com:389"/>
```



Agenda

- Why security?
- OC4J security
- Set up the user repository
- Set up web descriptor security
- Set up View layer security

Who is Running the App?

- Get user role from FacesContext

```
public boolean isAdmin() {
  FacesContext ctx =
                FacesContext.getCurrentInstance();
  ExternalContext ectx = ctx.getExternalContext();
  return (ectx.isUserInRole("admin"));
}
```

- This requires writing code in some utility class
- Alternative: use JSF-Security
  - Adds an EL scope: *securityScope*

JSF-Security

- Open source framework for exposing security settings to application
  - jsf-security.sourceforge.net
- Download library file and add it to the project
  - WEB-INF\lib
- Then role can be queried for value of properties on components
  - *Disabled*
  - *Rendered*
  - *Read-only*

Examples

- Hide container (af:tableSelectOne) for all but admin and manager roles

```
<af:tableSelectOne text="Select and"
    rendered=
   "#{securityScope.userInRole['admin,manager']}">
```

- Disable Salary item for all but admin roles

```
<af:inputText value="#{bindings.Salary.inputValue}"
   label="#{bindings.Salary.label}"
   required="#{bindings.Salary.mandatory}"
   columns="#{bindings.Salary.displayWidth}"
   disabled="#{ !securityScope.userInRole['admin']}"
/>
```

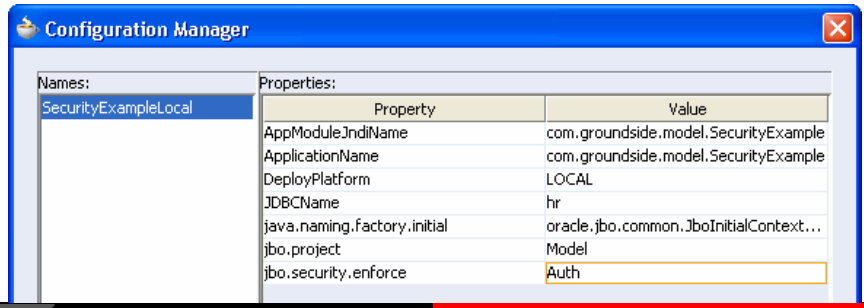
Demo

Securing ADF BC Attributes

- ADF BC can take the role of an authenticated user into account
- Used to secure entity attributes
  - Mark them as
    - Read-only
    - Updateable while new
    - Always Updatable
- Automatically reflected by the UI

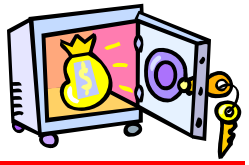
Steps to Secure Attributes

1. Tell ADF BC to worry about security
   - Set the configuration param **jbo.security.enforce=Auth**



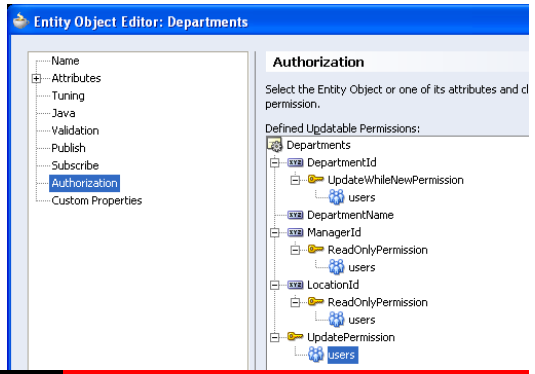
Steps to Secure Attributes

1. Tell ADF BC to worry about security
2. Propagate the JAZN-DATA.XML
   - Make sure that the following files contain the same users and roles:
     - %JDEV%/j2ee/home/config/system-jazn-data.xml
     - %JDEV%/jdev/system/oracle.j2ee.10.1.3.n.n/embedded-oc4j/config/system-jazn-data.xml
     - %workspace%/workspace-jazn-data.xml
   - This is just for design time



Steps to Secure Attributes

1. Tell ADF BC to worry about security
2. Propagate the JAZN-DATA.XML
3. Edit the Entity Object
   - Select the Authorization node



Demo

Other Techniques

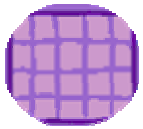
- Audit columns
  - Read user from app server session
  - Write to application context
  - Read the context in table DML triggers to assign CREATED_BY and MODIFIED_BY columns
- Method to preprocess query parameters
  - Use it to defeat SQL injection attempts
  - Process query criteria and strip out suspect characters

Other Resources

- *Declarative J2EE authentication and authorization with JAAS*
  - Frank Nimphius and Duncan Mills
  - www.oracle.com/technology/products/jdev/howtos/10g/jaassec/index.htm
- *Oracle Application Server Containers for J2EE Security Guide 10g Release 3 (10.1.3)*
  - download-east.oracle.com/docs/cd/B25221_04/web.1013/b14429/toc.htm
- Conquering the Fear Factor: Developing Secure J2EE Web Applications with Oracle ADF and JavaServer Faces, Frank Nimphius
  - Slides on the OOW website
- White paper for this talk
  - Hands-on practice
  - On the NoCOUG, OOW, and Quovera websites

Summary

- You need to design application security
- OC4J offers easy access to standard JAAS security features (JAZN)
- JAZN supports user repositories in XML and LDAP
- JDeveloper can help you define XML user repositories and hooks into the app
- Design and test for all security breach scenarios

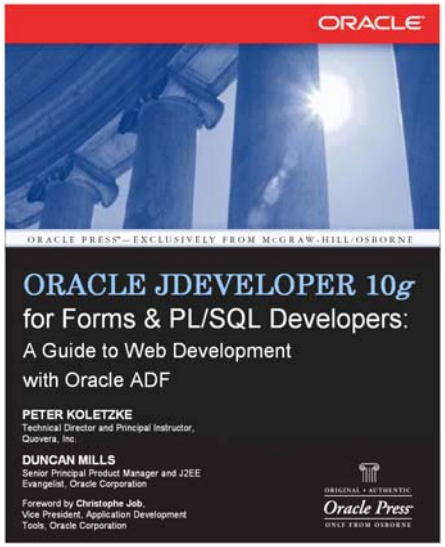


Out of Business

We will bankrupt ourselves
in the vain search
for absolute security.

—Dwight David Eisenhower, (1890-1969)

The Book



The Authors

- Peter Koletzke
  - Six other Oracle Press books about Oracle tools
  - www.quovera.com
- Duncan Mills
  - Widely published on OTN, ODTUG, etc.
  - groundside.com/blog/DuncanMills.php
  - www.oracle.com
- Book examples
  - www.tuhra.com